Please cancel claims 24, 47, and 57, and amend claims 6, 22, 25, 27, 42, 46, 48, and 56, as indicated below.

1. (Previously presented) A method in a computer system, the method comprising:

defining a plurality of transactionable locations, wherein individual ones of the transactionable locations encode respective values and are owned by no more than one transaction at any given point in a multithreaded computation;

for a particular non-blocking multi-target transaction of the multithreaded computation targeting two or more of the plurality of transactionable locations, attempting to acquire ownership of each of the transactionable locations targeted thereby, wherein the ownership acquiring wrests ownership from another non-blocking transaction that owns the targeted transactionable location without the other non-blocking transaction releasing ownership; and

once ownership of each of the targeted transactionable locations has been acquired, attempting to commit the particular non-blocking multi-target transaction using a single-target synchronization primitive to ensure that, at the commit, the particular non-blocking multi-target transaction continues to own each of the targeted transactionable locations, wherein individual ones of the non-blocking multi-target transactions do not contribute to progress of another.

2. (Previously presented) The method of claim 1,

wherein the ownership wresting employs a single-target synchronization primitive to change status of the wrested-from transaction to be incompatible with a commit thereof.

3. (Previously presented) The method of claim 2,

wherein, as a result of the status change, the wrested-from transaction fails and retries.

4. (Previously presented) The method of claim 2,

wherein the wrested-from non-blocking transaction is itself a non-blocking multi-target transaction.

5. (Original) The method of claim 1, further comprising:

on failure of the commit attempt, reacquiring ownership of each targeted transactionable location and retrying.

6. (Currently amended) The method of claim 1,

wherein no active transaction [[may]] is able to prevent another transaction from wresting therefrom ownership of transactionable locations targeted by the active transaction.

7. (Original) The method of claim 1,

wherein the ownership acquiring employs a single-target synchronization primitive to update the ownership of the targeted transactionable location.

8. (Original) The method of claim 1,

wherein each encoding of a transactionable location is atomically updateable using a single-target synchronization primitive.

9. (Original) The method of claim 1,

wherein the individual transactionable location encodings further include an identification of the owning transaction's corresponding value for the transactionable location.

10. (Original) The method of claim 1, further comprising:

accessing values corresponding to individual ones of the transactionable locations using a wait-free load operation.

11. (Original) The method of claim 1,

wherein the transactionable locations directly encode the respective values.

12. (Original) The method of claim 1,

wherein the transactionable locations are indirectly referenced.

13. (Previously presented) The method of claim 1,

wherein the transactionable locations are encoded in storage managed using a non-blocking memory management technique.

14. (Original) The method of claim 1,

wherein the transactionable locations, if unowned, directly encode the respective values and otherwise encode a reference to the owning transaction.

15. (Original) The method of claim 1,

wherein the single-target synchronization primitive employs a Compare-And-Swap (CAS) operation.

16. (Previously presented) The method of claim 1,

wherein the single-target synchronization primitive employs a Load-Linked (LL) and Store-Conditional (SC) operation pair.

17. (Original) The method of claim 1,

wherein the single-target of the single-target synchronization primitive includes at least a value and a transaction identifier encoded integrally therewith.

18. (Previously presented) The method of claim 1,

wherein the non-blocking multi-target transaction comprises a multi-target compare and swap (NCAS) operation.

19. (Previously presented) The method of claim 1,
embodied in operation of an application programming interface (API) that includes a load operation and a multi-target compare and swap (NCAS) operation.

20. (Original) The method of claim 19,
wherein the load operation is wait-free.

21. (Previously presented) The method of claim 1,
embodied in operation of an application programming interface (API) that provides transactional memory functionality.

22. (Currently amended) A computer-readable storage medium storing program instructions computer-executable to implement:

a plurality of non-blocking, multi-target transactions;

wherein the program instructions comprise:

instances of one or more single-target synchronization primitives executable to attempt to acquire, for a particular non-blocking multi-target transaction, ownership of two or more transactionable locations targeted by the non-blocking multi-target transaction so that ownership is wrested from respective other ones of the non-blocking multi-target transactions that own respective ones of the two or more targeted transactionable locations without the respective other ones of the non-blocking multi-target transactions releasing ownership; and

a particular single-target synchronization primitive executable to ensure that, at commit, the particular non-blocking multi-target transaction continues to own each of the two or more targeted transactionable locations; and

wherein individual ones of the non-blocking multi-target transactions do not contribute to progress of others.

23. (Previously presented) The storage medium of claim 22, wherein the program instructions are further executable to implement a concurrent computation, and wherein execution of the concurrent computation invokes the non-blocking multi-target transactions.

24. (Cancelled)

25. (Currently amended) The storage medium of claim [[24]] 22,

wherein to wrest ownership, the program instructions are further executable to implement an instance of a single-target synchronization primitive changing status of a wrested-from transaction to be incompatible with a commit thereof.

26. (Previously presented) The storage medium of claim 25,

wherein, as a result of the status change, the program instructions are further executable to implement the wrested-from transaction eventually failing and retrying.

27. (Currently amended) The storage medium of claim 22,

wherein no active transaction [[may]] is able to prevent another transaction from wresting therefrom ownership of transactionable locations targeted by the active transaction.

28. (Previously presented) The storage medium of claim 22,
wherein the two or more transactionable locations directly encode respective values.

29. (Previously presented) The storage medium of claim 22,
wherein the two or more transactionable locations are indirectly referenced.

30. (Previously presented) The storage medium of claim 22,
wherein the two or more transactionable locations are encoded in storage managed using a non-blocking memory management technique.

31. (Previously presented) The storage medium of claim 22,
wherein the two or more transactionable locations, if unowned, directly encode respective values and otherwise encode a reference to an owning transaction.

32. (Previously presented) The storage medium of claim 22,
wherein at least some instances of the one or more single-target synchronization primitives employ a Compare-And-Swap (CAS) operation.

33. (Previously presented) The storage medium of claim 22,
wherein at least some instances of the one or more single-target synchronization primitives employ a Load-Linked (LL) and Store-Conditional (SC) operation pair.

34. (Previously presented) The storage medium of claim 22,
wherein at least some of the non-blocking multi-target transactions comprise a multi-target compare and swap (NCAS) operation.

35. (Previously presented) The storage medium of claim 22,

wherein the program instructions comprise operations concurrently executable by one or more processors to operate on state of the two or more transactionable locations.

36. (Previously presented) The storage medium of claim 22,

wherein at least some of the non-blocking multi-target transactions are defined by an application programming interface (API) that includes a load operation and a multi-target compare and swap (NCAS) operation.

37. (Previously presented) The storage medium of claim 22,

wherein at least some of the non-blocking multi-target transactions are defined by an application programming interface (API) that provides transactional memory functionality.

38. (Previously presented) The storage medium of claim 22,

wherein the non-blocking multi-target transactions are obstruction-free, though not wait-free or lock-free.

39. (Previously presented) The storage medium of claim 22,

wherein the program instructions do not guarantee that at least one interfering concurrently executed non-blocking multi-target transactions makes progress.

40. (Previously presented) The storage medium of claim 22,

wherein the program instructions are further executable to implement a contention management facility configured to facilitate progress in a concurrent computation.

41. (Previously presented) The storage medium of claim 40,

wherein execution of the contention management facility ensures progress of the concurrent computation.

42. (Currently amended) The storage medium of claim 40,

wherein the contention management facility is modular such that employing alternative contention management strategies ~~may be employed without affecting~~ does not affect correctness.

43. (Previously presented) The storage medium of claim 40,

wherein the contention management facility allows changes in contention management strategy during a course of the concurrent computation.

44. (Cancelled)

45. (Cancelled)

46. (Currently amended) A computer readable storage medium storing program instructions computer-executable to implement:

instantiation of two or more transactionable locations in shared memory configured to individually encapsulate values that [[may be]] are targeted by concurrent executions of non-blocking multi-target transactions; and

one or more instances of a non-blocking multi-target transaction that upon execution of a particular instance thereof, attempts to acquire ownership of each of a plurality of transactionable locations targeted thereby and, once ownership of each of the plurality of targeted transactionable locations has been acquired, attempts to commit the particular instance using a single-target synchronization primitive to ensure that, at the commit, the particular instance continues to own each of the plurality of targeted transactionable locations[[,]];

wherein the ownership acquiring wrests ownership from another transaction that owns one of the plurality of targeted transactionable locations without the other transaction releasing ownership; and

wherein execution of no one of the non-blocking multi-target transaction instances contributes to progress of another.

47. (Cancelled)

48. (Currently amended) The storage medium of claim [[47]] 46,
wherein the another transaction is another concurrently executing instance of the non-blocking multi-target transaction.

49. (Previously presented) The storage medium of claim 46,
wherein at least some instances of the single-target synchronization primitive employ a Compare-And-Swap (CAS) operation.

50. (Previously presented) The storage medium of claim 46,
wherein at least some instances of the single-target synchronization primitive employ a Load-Linked (LL) and Store-Conditional (SC) operation pair.

51. (Previously presented) The storage medium of claim 46,
wherein the single-target of the single-target synchronization primitive includes a value and an owning transaction identifier encoded integrally therewith.

52. (Previously presented) The storage medium of claim 46,
wherein the program instructions are embodied as an application programming interface software component combinable with application program code to facilitate execution of the application program code as a multithreaded computation.

53. (Previously presented) The storage medium of claim 46,

wherein the non-blocking multi-target transaction implements a multi-target compare and swap operation (NCAS).

54. (Previously presented) The storage medium of claim 46,

wherein the non-blocking multi-target transaction implements transactional memory functionality.

55. (Previously presented) The storage medium of claim 46,

wherein the computer readable storage medium includes at least one medium selected from the set of a disk, a tape and another magnetic, optical, or electronic storage medium.

56. (Currently amended) An apparatus, comprising:

one or more processors;

one or more data stores addressable by each of the one or more processors; and

means for coordinating concurrent non-blocking execution, by the one or more processors, of non-blocking multi-target transactions that attempt to acquire ownership of each of a plurality of transactionable locations targeted thereby and, once ownership of each of the plurality of targeted transactionable locations has been acquired, attempt to commit a particular instance thereof using a single-target synchronization primitive to ensure that, at the commit, the particular instance continues to own each of the plurality of targeted transactionable locations, wherein the ownership acquiring wrests ownership from another transaction that owns one of the plurality of targeted transactionable locations without the other transaction releasing ownership, and wherein none of the non-blocking multi-target transaction contributes to progress of another.

57. (Cancelled)

58. (Previously presented) The apparatus of claim 56,

wherein the wresting means includes means for ensuring that status of the wrested-from transaction is incompatible with a successful commit thereof.

59. (Previously presented) The apparatus of claim 56, further comprising:

means for managing contention between interfering executions of the non-blocking multi-target transactions.